An Adaptive Algorithm for Modifying Hyperellipsoidal Decision Surfaces

Patrick M. Kelly*, Don R. Hush*, James M. White**

*Department of Electrical and Computer Engineering,
University of New Mexico, Albuquerque, New Mexico 87131

**Computer Research and Applications Group C-3, MS B265,
Los Alamos National Laboratory, Los Alamos, New Mexico 87545

Abstract

The LVQ algorithm is a common method which allows a set of reference vectors for a distance classifier to adapt to a given training set. We have developed a similar learning algorithm, LVQ-MM, which manipulates hyperellipsoidal cluster boundaries as opposed to reference vectors. Regions of the input feature space are first enclosed by ellipsoidal decision boundaries, and then these boundaries are iteratively modified to reduce classification error. Results obtained by classifying the Iris data set are provided.

1 Introduction

The classifier developed in this paper combines concepts from both the LVQ [1] and the RCE [2] classification methods. It is similar to the RCE network in that it uses a thresholded distance metric, but the distance metric is a Mahalanobis distance [3]. This allows the classifier to partition the pattern space into hyperellipsoidal decision regions (giving rise to piecewise quadratic decision boundaries in overlap regions). The learning algorithm that we develop is similar to the LVQ algorithm, except that it must adjust more than just the positions of the reference vectors. In this respect our approach is similar to the approach in [4] where parameters other than just the reference vectors are adjusted for the RBF network. We refer to the learning algorithm developed in this paper as the LVQ-MM algorithm (LVQ using the Mahalanobis distance Metric).

2 Classifier Initialization

Our classifier works as follows. A single class of data is represented by several hyperellipsoidal clusters. Data falling inside of at least one hyperellipsoid is classified as in-class data. A hyperellipsoid is defined by a mean vector, $\underline{\mu}$, a covariance matrix (symmetric and positive definite), Σ, and an effective radius, d. The mean vector determines the location of the hyperellipsoid, while the covariance matrix determines its shape and orientation. The squared Mahalanobis distance [3] is used to determine if a vector $\underline{x}$ lies inside or outside of a given hyperellipsoid. Vectors lying inside of a hyperellipsoid satisfy:

$$(\underline{x} - \underline{\mu})^T \Sigma^{-1} (\underline{x} - \underline{\mu}) < d \tag{1}$$

An initial clustering of the training data for each class is needed as the basis for this classifier. There are numerous ways in which this might be accomplished. We use the method proposed in [5] which suggests using the k-means clustering algorithm [3] followed by a cluster merging process. After clusters have been initialized in this way, effective radii are selected for each cluster in the set. We choose the effective radius d for each cluster so that under the assumption that the data is gaussian in nature, P% of the data will

fall inside the boundary (typically P=99%). For an N-dimensional problem, the distribution of the squared Mahalanobis distances to each of the vectors within a given cluster is a χ^2 distribution with N degrees of freedom. If pattern vectors consist of 21 features, for example, then an effective cluster radius of 38.9 would cause the hyperellipsoidal boundaries to contain 99% of the data.

3 Classifier Adaptation

After classifier initialization, each class of data is represented by a number of hyperellipsoidal clusters in the pattern space. An adaptive training algorithm is now employed to reduce classification error in areas of known overlap between different classes. During adaptation, each hyperellipsoid in the classifier will maintain its original orientation, although its position and shape will be modified. The algorithm that we use is similar to the LVQ algorithm, and thus is referred to as the LVQ-MM algorithm (LVQ with the Mahalanobis distance Metric).

Assume that we have a hyperellipsoidal decision boundary, and a new vector which we would like to include within the in-class data region (see Figure 1). We are only going to allow the mean vector $\underline{\mu}$ and the eigenvalues of the covariance matrix Σ to change. This means that the orientation of the cluster will remain fixed, although its position and shape may be modified. The cluster will be modified in such a way that the following are true: (1) the new point, $\underline{x}_n$, lies on the new cluster boundary; (2) the original boundary point lying on the opposite side of the hyperellipsoid from $\underline{x}_n$ remains on the cluster boundary; and (3) the eigenvalues of Σ are modified in such a way that the hyperellipsoid is only stretched "towards" $\underline{x}_n$.

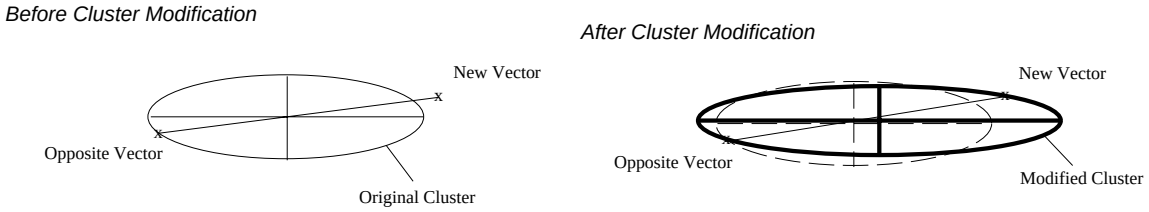

Figure 1: Modifying a Hyperellipsoidal Boundary

The new mean vector for the cluster will lie directly between $\underline{x}_n$ and the point lying opposite it on the hyperellipsoidal boundary, and is given by:

$$\underline{\mu}_n = \frac{1}{2}\left[\left(1+\sqrt{\frac{d}{m_n}}\right)\underline{\mu} + \left(1-\sqrt{\frac{d}{m_n}}\right)\underline{x}_n\right] \tag{2}$$

After computing $\underline{\mu}_n$, we need to modify Σ. In doing so, we are essentially going to stretch the decision boundary towards the new point, $\underline{x}_n$. Note that if the new point lies along one of the hyperellipsoidal axes, that only one eigenvalue for Σ will need to be changed. Otherwise, all (or at least several) eigenvalues will need to be modified. For simplicity, we will restrict our attention to modifying the inverse of Σ. Let us decompose Σ^{-1} as follows:

$$\Sigma^{-1} = M\Lambda M^T \tag{3}$$

where M is the orthonormal eigenvector matrix for Σ^{-1}, and Λ is the diagonal eigenvalue matrix. To stretch the cluster we will modify the eigenvalues of Σ^{-1}, and keep the eigenvectors fixed. Thus we wish to find a new inverse covariance matrix:

$$\Sigma_n^{-1} = M\Lambda\Lambda_\delta M^T \tag{4}$$

where the "stretch matrix", Λ_δ, takes on the form:

$$\Lambda_\delta = \begin{bmatrix} 1+\delta_1 p & 0 & \cdots & 0 \\ 0 & 1+\delta_2 p & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & 1+\delta_N p \end{bmatrix} \tag{5}$$

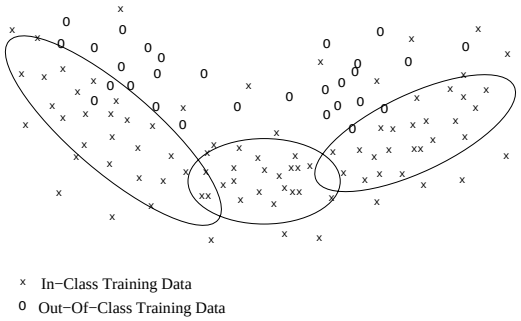

Figure 2: A Two-Dimensional System Before Adaptation

The parameter p determines the total stretch, and the parameters δ_i determine the percentage stretch in the direction of the i^{th} principal component. The δ_i satisfy the following constraints:

$$0 \leq \delta_i \leq 1, \qquad \sum_{i=1}^{N} \delta_i = 1 \tag{6}$$

Our goal is to determine the parameters p and δ_i so that the hyperellipsoidal boundary is moved to $\underline{x}_n$:

$$(\underline{x}_n - \underline{\mu}_n)^T \Sigma_n^{-1} (\underline{x}_n - \underline{\mu}_n) = (\underline{x}_n - \underline{\mu}_n)^T M \Lambda \Lambda_\delta M^T (\underline{x}_n - \underline{\mu}_n) = d \tag{7}$$

Let us define a new vector $\underline{z}$ to be:

$$\underline{z} = \Lambda^{0.5} M^T (\underline{x}_n - \underline{\mu}_n) \tag{8}$$

The components of this vector, z_i, $i = 1, 2, ..., N$, represent the strength of the projection of $\underline{x}_n$ onto the N principal components. With this, (7) can be rewritten as:

$$\underline{z}^T \Lambda_\delta \underline{z} = \sum_{i=1}^{N} z_i^2 (1 + \delta_i p) = d \tag{9}$$

The percentages, δ_i, are chosen to be equal to the relative magnitude of the projection of $\underline{x}_n$ onto each of the principle components:

$$\delta_i = \frac{\mid z_i \mid}{\sum_{j=1}^{N} \mid z_j \mid} \tag{10}$$

It is easy to verify that this choice for δ_i satisfies (6). Substituting (10) into (9) and solving for p we get:

$$p = (d - \hat{m}_n) \, \frac{\sum_{i=1}^{N} \mid z_i \mid}{\sum_{i=1}^{N} \mid z_i \mid^3} \tag{11}$$

where $\hat{m}_n$ is the squared Mahalanobis distance to $\underline{x}_n$ using the original covariance matrix Σ and our new mean vector $\underline{\mu}_n$.

In summary, the new inverse covariance matrix is given by (4) where the components of Λ_δ are computed using (8), (10), and (11).

The cluster modification method discussed above provides a foundation for the classifier adaptation algorithm. Using the cluster modification technique to move clusters "toward" and "away from" training vectors, this algorithm will attempt to minimize classification error in regions of known overlap between classes. We will restrict our attention to modifying clusters for a single class of data only. This process is then used independently for each class of data.

4 The LVQ-MM Algorithm

Consider the two-dimensional problem shown in Figure 2. The classifier has been initialized using the in-class training data, and consists of three hyperellipsoids in the input space. Notice that based on the training data available to the classifier, there is no overlap in the region "below" the current hyperellipsoids. The only conflicting areas lie on the upper portion of the closed decision region. This suggests that better classifier performance can be achieved by allowing the decision region to include all in-class vectors lying below the current set, and by moving the upper boundary of the classifier to a position minimizing misclassifications in that area.

A single step in the LVQ-MM algorithm will basically work as follows. Select a random vector from the training data which is currently misclassified (correctly classified samples do not affect the classifier training). Using the cluster adaptation equations previously derived, move one of the hyperellipsoidal boundaries either toward or away from this vector, depending on its class membership. Note that as the cluster adaptation equations currently stand, this step will always cause the current vector to fall directly on the new hyperellipsoidal boundary.

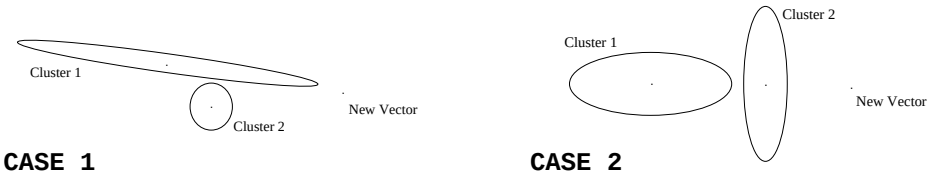

Figure 3: Selecting Desired Boundary

The question to be addressed now is, "Which of the hyperellipsoids should be modified?" Consider the two cases illustrated in Figure 3. In Case 1, we want to modify cluster 1 to include the new vector even though the mean vector for cluster 1 is farther away from the new vector than the mean vector for cluster 2 in terms of Euclidean distance. In terms of Mahalanobis distance, however, the opposite is true. In Case 2, on the other hand, the roles are reversed. We want to modify cluster 2 to include the new vector. The Euclidean distance to cluster 2 is smaller than the Euclidean distance to cluster 1, and in terms of Mahalanobis distance, cluster 1 is closer. Clearly, the cluster to be modified should be the cluster whose boundary is closest to the new vector (in terms of Euclidean distance). It can be shown that the distance from the boundary to the new vector is given by

$$dist = |(1 - \sqrt{\frac{d}{m}})| \cdot \|\underline{x}_n - \mu\| \tag{12}$$

where d is the effective cluster radius, and m is the squared Mahalanobis distance to the new vector. The adaptation loop, then, works as follows.

LVQ-MM ALGORITHM

(1) Select a training vector that is misclassified
(2) Determine which cluster to modify
(3) Modify mean using (2)
(4) Modify inverse covariance matrix using (4)

This algorithm is typically run for several passes through the training data.

5 Experimental Results

The Iris data set[1] [6] was used by R. A. Fisher in 1936 to discuss the use of linear discriminant functions. The set contains 50 four-dimensional vectors from each of three different classes of flowers: Iris setosa, Iris versicolor, and Iris virginica. The Iris setosa data is linearly separable from the the other two classes, but the Iris versicolor and Iris virginica data are not linearly separable from one another.

Since the Iris setosa data is easily separated from the other two classes of data, we directed our work towards identifying Iris versicolor and Iris virginica data. For a given class of data (Iris versicolor or Iris virginica) containing 50 sample vectors, 25 were selected as training vectors and the other 25 were used as a test set to determine how well the classifier generalizes to new data. Because discriminating between these classes is highly dependent upon the training sets used, ten different training/test sets were chosen at random, and the results provided reflect average classifier performance.

As a basis for comparison we trained a linear classifier to discriminate between Iris versicolor and Iris virginica data. The perceptron learning algorithm [3] was used. The overall results of the linear classification are shown in Table 1.

	Correct	Incorrect
Training Data	96.0%	4.0%
Test Data	91.0%	9.0%

Table 1: Results Using the Perceptron Learning Algorithm

Using a linear classifier for this problem seems to work fairly well. When presented with Iris setosa data, however, the classifier will respond as if it were presented with Iris versicolor data. We next applied our our one-class classification scheme to this data. Ignoring the Iris setosa data, we attempted to build a one-class classifier for each of the other two sets of data. Because the data is known to be unimodal, a single cluster (μ, Σ) was estimated for each class. An effective cluster radius (d) of 14.9 was chosen. For a four-dimensional problem with a gaussian distribution, this effective radius reflects a confidence interval containing 99.5% of the data. Given a single cluster, then, classification results were computed for both the training data and the test data. In order to improve the classifier performance, the LVQ-MM algorithm was run using both the in-class and out-of-class training data. The adaptation process was halted after 50 passes through the training data.

Using the single cluster for the Iris versicolor training data (before adaptation) classified nearly all of the in-class data correctly (see Table 2). Out-of-class data (Iris virginica), however, were frequently incorrectly classified as in-class (Iris versicolor) data. Results for rejecting out-of-class data were significantly improved after the LVQ-MM algorithm was employed. After this adaptation process, only 0.4% of all training data was classified incorrectly, while 6.2% of all test data was classified incorrectly. Unlike the results obtained with the linear classifier, all 50 vectors from the Iris setosa data were rejected by this classifier both before and after adaptation. Similar results were obtained using the Iris virginica classifier (see Table 3).

	Before Adaptation		After Adaptation	
	In-Class	Out-Of-Class	In-Class	Out-Of-Class
Training Data	100.0%	77.6%	99.6%	99.6%
Test Data	99.2%	80.4%	90.8%	96.8%

Table 2: Correct Classifications Using Versicolor One-Class Classifier

The LVQ-MM learning algorithm attempts to move the decision boundary away from the out-of-class data which it misclassifies. The result is that more data overall is rejected when using the adapted classifier

[1] We would like to acknowledge the assistance received by using the UCI Repository Of Machine Learning Databases and Domain Theories.

	Before Adaptation		After Adaptation	
	In-Class	Out-Of-Class	In-Class	Out-Of-Class
Training Data	100.0%	65.2%	93.6%	99.6%
Test Data	94.0%	65.0%	81.6%	93.6%

Table 3: Correct Classifications Using Virginica One-Class Classifier

as compared to the amount of data rejected before classifier adaptation. Table 4 shows the percentage of Iris versicolor and Iris virginica data which was accepted and rejected by the two classifiers. Very little data falls into the wrong cluster, but there is a substantial amount of data that falls into both clusters before classifier adaptation. After adaptation, the overlap region has been greatly reduced, and as a result the number of vectors rejected from both classifiers has increased.

	Before Adaptation				After Adaptation			
	Right	Wrong	Both	Rejected	Right	Wrong	Both	Rejected
Training Data	71.4%	0.0%	28.6%	0.0%	97.0%	0.0%	0.6%	2.4%
Test Data	70.0%	0.2%	26.6%	3.2%	84.4%	3.0%	1.8%	10.8%

Table 4: Overview of Data Classifications

The one-class classifiers worked well in the sense that they always rejected data that was dissimilar from the training data used for classifier design. The Iris setosa data, which is misclassified when using the linear classifier, is correctly classified as "out-of-class data" when using the one-class classification approach. As for the discrimination capability of the classifier, the adapted Iris versicolor one-class classifier actually performed better than the linear classifier did for these data sets. The Iris virginica classifier, on the other hand, performed slightly worse than the linear classifier.

6 Conclusions

Statistical methods of pattern recognition have been used extensively for many years. Using the method presented in this paper decision boundaries can be manipulated to reduce classification error. Our method has the combined advantage of minimizing classification error between classes for which training data is available, while at the same time rejecting patterns from other classes which are dissimilar to the training data.

References

[1] T. Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, Berlin, Germany, 1988.

[2] D.L. Reilly, L.N. Cooper, and C. Elbaum. A neural model for category learning. *Biological Cybernetics*, 45:35–41, 1982.

[3] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, NY, 1973.

[4] S. Lee and R. Kil. A gaussian potential function network with hierarchically self-organizing learning. *Neural Networks*, 4:207–224, 1991.

[5] P.M. Kelly. *A One-Class Classifier Using Hyperellipsoidal Decision Surfaces.* Masters Thesis, University of New Mexico, 1991.

[6] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.